Terms of Reference

Reusable Component: Document Upload & Attachment Framework

Mendix Reusable Components Initiative - ISAA

Version	Status
0.0.1	Draft

1. Background

This Terms of Reference document is issued as part of ISAA's (Information Systems Agency of Armenia) initiative to build a standard, reusable foundation of Mendix components for public digital services. Full context, goals, and target users for the overall initiative are described in the accompanying **Concept Note**.

Nearly every public service asks the citizen or business to supply documents - a certificate, a floor plan, a diploma, a signed consent. Mendix provides file storage primitives, but each project currently builds its own layer on top of them: its own rules about acceptable file types and sizes, its own malware handling, its own definition of which documents a service requires, its own access rules and its own retention behaviour. This component addresses that gap by providing that layer once, scoped to the **capture, custody and controlled release** of documents. Reading the contents of a document is the subject of a separate Terms of Reference, the **Document OCR, Extraction & Interpretation Framework**, which consumes this component rather than duplicating it.

2. Objective

Develop a configurable, reusable **Document Upload & Attachment Framework** for Mendix applications that standardizes uploading, validating, storing, securing, presenting and retiring documents attached to business objects across public digital services.

3. Scope of Work

3.1 Functional Requirements

Standardized Attachment Model

Provide a common representation of an attached document - its type, its origin, its metadata and its state - regardless of the business object it belongs to or the service that captured it.

Reusable Attachment Capabilities

The framework should provide reusable capabilities for:

- Document type configuration and completeness checking;
- Upload and capture, including mobile devices;
- Validation and malware scanning;
- Storage, integrity and preview;
- Access control on the file itself;
- Retention, legal hold and purge;
- Search and administration.

Attachment to Any Business Object

An application must be able to attach documents to its own entities without that application's domain model and this component becoming structurally dependent on one another.

Configurable Document Type Catalogue

An administrator should be able to define, per service, which document types are expected - their name, whether they are mandatory, how many may be supplied, and which formats and size limits apply - without development work. The framework should expose a reusable completeness check answering "has everything required been supplied?" so that each application does not implement that question itself.

Upload Experience

Provide reusable upload capabilities covering single and multiple files, progress indication, clear and specific rejection messages, capture on mobile devices (including photographing a paper document with the device camera), and replacement of a document uploaded in error.

Validation

File type, size and count limits must be enforced on the server, not only in the browser, and must be driven by the document type configuration rather than hard-coded. The framework should verify that a file's actual content matches its declared type, so that an executable renamed to a permitted extension is rejected.

Malware Scanning

Uploaded files must be scanned before they become available to any officer or downstream

component. A file that fails scanning must be quarantined or destroyed according to configuration, and the outcome recorded.

Storage Abstraction

The framework should not assume a single storage medium. The storage back end (e.g., local file system, object storage, or a government-hosted equivalent) should be abstracted so that changing it does not affect adopting applications.

Integrity

Record a cryptographic checksum for every stored document, so that a document produced years later as evidence can be shown to be the one that was submitted.

Preview & Download

Officers should be able to view common document formats in the browser without downloading them to a workstation, with download available where permitted.

Authorization/Access

Provide a reusable authorization model that integrates with the application's security model, controlling which attachments a user may list and preview, and which they may download, replace or delete, without application-specific authorization logic. Access must be enforced on the file itself, not only on the page that lists it - a document must not be retrievable by anyone holding or guessing its URL.

Retention, Legal Hold & Purge

Support configurable retention periods per document type, with an archive-then-purge mechanism, and the ability to place a hold that suspends deletion where a case is under dispute or investigation.

Search & Administration

Support locating attachments by owning object, document type, uploader and date range, and provide administrative operations for bulk export and for correcting mis-filed documents.

Integration with Other Reusable Components

The framework should be usable by, and integrate with, other components in this initiative - in particular the **Application & Case Management Framework** (documents belonging to a case, and completeness as a condition of submission), the **Audit & Activity History Framework** (upload, replacement, download and deletion appearing in the activity history), the **Document OCR, Extraction & Interpretation Framework**, which must be able to obtain a stored document through a defined interface rather than reaching into storage directly, and the **API Integration Framework** where a document is received from or delivered to an external system. Integration points and mechanisms should be reviewed and defined together with each relevant component's vendor.

Configuration-Based Adoption

New document types and service-specific document requirements should be enabled through configuration where possible, minimizing custom development effort.

Vendors should describe:

- How a new service defines its document types and requirements (configuration steps vs. required development), how an attachment refers to its owning object, and what an adopting application must and must not change in its own model;
- Their approach to large files and poor connectivity - whether uploads are resumable, and what the applicant sees when an upload fails part-way - and whether a superseded file is retained after replacement;
- The malware scanning mechanism, whether scanning is synchronous or deferred, what the applicant and the officer see while a file is pending scan, and who is notified of a failure;
- How the storage back end is abstracted and what a migration between back ends would involve;
- Which formats are previewable in the browser and how unsupported formats are handled;
- How direct file access is prevented and how access decisions are enforced regardless of path.

3.2 Non-Functional Requirements

- **Security**: Access, scanning and integrity rules enforced consistently regardless of access path (UI, API, direct file request, scheduled process); documents protected in transit and at rest.
- **Performance**: Uploads, scanning and preview generation must not block transactional business microflows; heavy operations should run asynchronously.
- **Scalability**: Storage growth manageable over years of operation; listing and searching attachments must remain performant for objects with many documents.
- **Observability**: Structured logging of upload, scan, access and deletion events sufficient to support the Audit & Activity History Framework and operational troubleshooting.
- **Compatibility**: Must run on *Mendix version > 11.10.0* and be upgradable across supported Mendix versions.
- **Localization**: Document type names, guidance and rejection messages maintainable in Armenian and English.

4. Deliverables

Vendors are expected to provide the following, in addition to the specific requirements above:

Standard Implementation Deliverables

#	Deliverable	Description
1	Mendix Source	Mendix application source (Team Server/Git) and deployment package.
2	Technical Documentation	Module purpose, architecture, dependencies, exposed entities, microflows, Java actions, and configuration.
3	Architecture Documentation	Architecture overview and key design decisions.
4	Security Configuration	Module roles, user role mappings, authentication and authorization configuration.
5	API Documentation	Interface specifications, request/response schemas, sample payloads, and error handling.
6	Configuration Guide	Environment-specific constants and configuration settings.
7	Third-Party Component Inventory	Marketplace modules and external dependencies used.
8	Code Quality Report	Static code analysis and Mendix quality metrics.
9	Test Plan	Unit, integration, system, UAT, regression, and non-functional testing approach.
10	Test Results	Test cases and execution evidence.
11	Security Test Report	Security testing and vulnerability assessment results (where applicable).
12	Performance Test Report	Performance and load testing results (where applicable).
13	User Documentation	User and administrator guides.
14	Training Materials	Training guides and supporting materials (where applicable).

Additional Deliverables for Reusable Components

#	Deliverable	Description
1	Module README	Purpose, prerequisites, installation and upgrade instructions.

2	Interface Contract	Public microflows, entities, Java actions, parameters, and expected outputs, including the interface by which another component obtains a stored document.
3	Configuration Guide	Post-import configuration instructions, including storage back end and malware scanning setup.
4	Sample Application	Reference application demonstrating component implementation, including a configured document type catalogue with mandatory and optional types, a completeness check, a rejected upload (wrong type and oversized), and a document retrieved through the public interface by a consuming component.

5. Vendor Response Requirements

Vendor proposals should address:

1. **Technical approach** - proposed architecture, how it maps to the functional requirements in Section 3, and how it builds on (rather than replaces) Mendix's native file handling.
2. **Configuration model** - concretely how a new service defines its document types and requirements, and what is/isn't possible without custom development.
3. **Storage & integrity approach** - storage abstraction, checksums, and what a migration between storage back ends would involve.
4. **Security approach** - malware scanning mechanism, authorization model, and how direct file access is prevented.
5. **Retention approach** - retention configuration, legal hold, and archive/purge mechanism.
6. **Reuse & extensibility** - how the framework avoids tight coupling to any single adopting application, and how it will be consumed by other components in this initiative, particularly the OCR component.
7. **Team & experience** - Experience with Mendix (certificates, if available) and completed relevant projects, in particular document management or content-storage systems.
8. **Timeline** - proposed delivery schedule against the deliverables in Section 4.
9. **Indicative Cost** - cost breakdown by phase/deliverable.
10. **Assumptions & risks** - any assumptions made and risks identified.